

1 What is claimed is:

2

3 1. A file server comprising storage containing a file system, and a processor coupled
4 to the storage for accessing the file system, wherein the file system includes a production
5 file, read-only snapshot copies of the production file, and at least one read-write snapshot
6 copy of the production file, wherein the production file and the snapshot copies of the
7 production file are organized as a version set including an inode for the production file
8 and an inode for each snapshot copy of the production file, and a set of file blocks
9 including data blocks and indirect blocks that are shared among the production file and
10 the snapshot copies of the production file.

11

12 2. The file server as claimed in claim 1, which is programmed to create a new fully
13 preallocated production file by allocating an inode for the production file, allocating to
14 the production file data blocks for a specified size of the production file, and allocating to
15 the production file any and all indirect blocks needed to link the data blocks of the
16 production file to the inode of the production file.

17

18 3. The file server as claimed in claim 1, which is programmed to create a new sparse
19 production file by initially allocating just an inode for the production file, and wherein
20 the file server is programmed to allocate other blocks for the sparse production file as
21 needed when data blocks of the sparse production file are written to.

22

1 4. The file server as claimed in claim 1, wherein the version set includes a version
2 chain linking the inode for the production file to the nodes for the read-only snapshot
3 copies of the production file.

4
5 5. The file server as claimed in claim 4, wherein the inode for each read-write
6 snapshot copy of the production file is linked to a corresponding one of the nodes for the
7 read-only snapshot copies of the production file.

8
9 6. The file server as claimed in claim 1, wherein each inode in the version set
10 includes a version pointer field and a branch pointer field, contents of the version pointer
11 fields link the nodes of the read-only snapshot copies of the production file into the
12 version set, and contents of the branch pointer fields link each inode of each read-write
13 snapshot copy of the production file into the version set.

14
15 7. The file server as claimed in claim 6, wherein the contents of the version pointer
16 fields form a version chain linking the inode of the production file to nodes of the read-
17 only copies of the production file.

18
19 8. The file server as claimed in claim 7, wherein the contents of the branch pointer
20 fields link each inode of each read-write snapshot copy of the production file system to
21 an inode of a respective one of the read-only snapshot copies of the production file.

22

1 9. The file server as claimed in claim 1, wherein the file server is programmed for
2 parsing path names through the version set to locate the nodes in the version set, each
3 path name including a name for the production file, and each path name for a snapshot
4 copy of the production file includes the name for the production file followed by a
5 delimiter symbol followed by a name for the snapshot copy of the production file.

6
7 10. The file server as claimed in claim 9, wherein each path name for a read-write
8 snapshot copy of the production file includes the name for the production file followed by
9 a delimiter symbol followed by a name for a read-only snapshot copy of the production
10 file followed by a delimiter symbol followed by a name for the read-write snapshot copy
11 of the production file.

12
13 11. The file server as claimed in claim 1, wherein the file server is programmed for
14 creating a new read-only snapshot copy of the production file, creating a new read-write
15 snapshot copy of the production file, deleting a snapshot copy of the production file from
16 the version set, restoring the production file with a specified snapshot copy of the
17 production file, refreshing a specified snapshot copy of the production file, and naming
18 the files in the version set.

19
20 12. The file server as claimed in claim 1, which is programmed for creating a new
21 read-write snapshot copy of a specified file in the version set by allocating an inode for
22 the new read-write snapshot copy, copying the inode of the specified file to the inode for

1 the new read-write snapshot copy, and designating that blocks of the specified file are
2 shared with the new read-write snapshot copy.

3
4 13. The file server as claimed in claim 12, wherein the nodes contain pointer fields
5 for pointers to the file blocks, and wherein the file server is programmed to designate that
6 blocks of the specified file are shared with the new read-write snapshot copy by setting
7 flags in the pointer fields in the inode for the read-write snapshot copy.

8
9 14. The file server as claimed in claim 1, which is programmed to maintain for each
10 block in each read-only snapshot copy of the production file an indication of whether or
11 not said each snapshot copy of the production file is an oldest snapshot copy of the
12 production file including an identical version of said each block.

13
14 15. The file server as claimed in claim 14, wherein the nodes in the version set
15 include pointer fields, and wherein each pointer field includes a pointer to a block in the
16 version set and a flag for indicating whether or not the version of the production file of
17 the inode is an oldest snapshot copy of the production file including an identical version
18 of the block pointed to by the pointer in said each pointer field.

19
20 16. The file server as claimed in claim 14, wherein each inode of each read-only
21 snapshot copy of the production file is linked to a hierarchy of blocks included in said
22 each read-only snapshot copy of the production file, the inode of said each read-only
23 snapshot copy of the production file includes an indication of whether or not said each

1 read-only snapshot copy of the production file is an oldest read-only snapshot copy of the
2 production file including an identical version of each block that is a child of said each
3 inode in the hierarchy of blocks included in said each read-only snapshot copy of the
4 production file, and when said each read-only snapshot copy of the production file is not
5 an oldest read-only snapshot copy of the production file including an identical version of
6 said each block that is a child of said each inode in the hierarchy the of blocks, then said
7 each read-only snapshot copy of the production file is not an oldest read-only snapshot
8 copy of the production file including an identical version of each descendant of said each
9 block that is a child of said each inode in the hierarchy of the blocks.

10
11 17. The file server as claimed in claim 14, which is programmed to delete a read-only
12 snapshot copy of the production file, and when deleting the read-only snapshot copy of
13 the production file, to keep each block for which the read-only snapshot copy is not
14 indicated as being an oldest snapshot copy of the production file including an identical
15 version of said each block.

16
17 18. The file server as claimed in claim 17, wherein each inode of each read-only
18 snapshot copy of the production file is linked to a hierarchy of blocks included in said
19 each read-only snapshot copy of the production file, and wherein the file server is further
20 programmed to keep all descendants of said each block for which the read-only snapshot
21 copy is not indicated as being an oldest snapshot copy of the production file including an
22 identical version of said each block.

23

1 19. The file server as claimed in claim 14, which is programmed to delete a read-only
2 snapshot copy of the production file, and when deleting the read-only snapshot copy of
3 the production file, to keep each block for which the read-only snapshot copy of the
4 production file is indicated as being an oldest snapshot copy of the production file
5 including an identical version of said each block and a next-most recent version of the
6 production file is indicated as not being an oldest snapshot copy of the production file
7 including an identical version of said each block.

8
9 20. The file server as claimed in claim 19, which is further programmed, upon
10 deleting the read-only snapshot copy of the production file, to indicate that the next-most
11 recent snapshot copy of the production file has become an oldest snapshot copy of the
12 production file including an identical version of said each block for which the read-only
13 snapshot copy of the production file is indicated as being an oldest snapshot copy of the
14 production file including an identical version of said each block and a next-most recent
15 snapshot copy of the production file is indicated as not being an oldest snapshot copy of
16 the production file including an identical version of said each block.

17
18 21. The file server as claimed in claim 14, which is programmed to delete a read-only
19 snapshot copy of the production file by deallocating each block for which the read-only
20 snapshot copy is indicated as an oldest read-only snapshot copy of the production file
21 including an identical version of said each block and a next most recent read-only
22 snapshot copy of the production file is also indicated as an oldest read-only snapshot
23 copy of the production file including an identical version of a block corresponding to said

1 each block, wherein said each block and the block corresponding to said each block are
2 mapped to the same logical file addresses.

3

4 22. The file server as claimed in claim 1, which is programmed for responding to a
5 request to create a read-only snapshot copy of the production file by reserving for the
6 production file a number of free file blocks of at least the number of blocks in the
7 production file.

8

9 23. The file server as claimed in claim 1, which is programmed for responding to a
10 request to create a read-write snapshot copy of a specified file in the version set by
11 reserving for the read-write snapshot copy a number of free blocks of at least the number
12 of blocks in the specified file in the version set.

13

14 24. The file server as claimed in claim 1, which is programmed for responding to a
15 request to restore the production file with a specified snapshot copy of the production file
16 by reserving for the production file a number of free blocks of at least the difference
17 between the number of blocks in the specified snapshot copy of the production file and
18 the number of blocks in the production file.

19

20 25. The file server as claimed in claim 1, which is programmed for restoring the
21 production file with a specified snapshot copy of the production file by responding to a
22 request to prepare to restore the production file by preparing to restore the production file
23 and reporting whether or not preparation is successful, and then responding a request to

1 commit the preparation by restoring the production file with the specified snapshot copy
2 of the production file.

3

4 26. The file server as claimed in claim 25, which is programmed to prepare to restore
5 the production file by creating a read-write snapshot copy of the specified snapshot copy
6 of the production file, and which is programmed to commit the preparation by replacing
7 the production file with the read-write snapshot copy of the specified snapshot copy of
8 the production file and deleting the production file, so that the read-write snapshot copy
9 of the specified snapshot copy of the production file assumes the identity of the
10 production file.

11

12 27. The file server as claimed in claim 1, which includes refreshing a specified
13 snapshot copy of the production file by creating a new inode in the version set, copying
14 contents of the inode of the specified snapshot copy into the new inode so that the new
15 inode references blocks of the specified snapshot copy of the production file, using the
16 inode of the specified snapshot copy to create a new snapshot copy of the production file
17 by copying contents of the inode of the production file into the inode of the specified
18 snapshot copy, and performing a file deletion upon the new node.

19

20 28. The file server as claimed in claim 27, which is programmed to perform the file
21 deletion upon the new inode asynchronously after using the inode of the specified
22 snapshot copy to create a new snapshot copy of the production file by copying contents
23 of the inode of the production file into the inode of the specified snapshot copy.

1
2 29. A file server comprising storage containing a file system, and a processor coupled
3 to the storage for accessing the file system, wherein the file system includes a production
4 file, read-only snapshot copies of the production file, and at least one read-write snapshot
5 copy of the production file, wherein the production file and the snapshot copies of the
6 production file are organized as a version set including an inode for the production file
7 and an inode for each snapshot copy of the production file, and a set of file blocks
8 including data blocks and indirect blocks that are shared among the production file and
9 the snapshot copies of the production file, wherein the file server further includes:

10 means for creating new read-only snapshot copies of the production file;

11 means for creating new read-write snapshot copies of the production file;

12 means for deleting a specified snapshot copy of the production file from the
13 version set;

14 means for restoring the production file with a specified snapshot copy of the
15 production file;

16 means for refreshing a specified snapshot copy of the production file; and

17 means for naming the files in the version set.
18

19 30. A file server comprising storage containing a file system, and a processor coupled
20 to the storage for accessing the file system, wherein the file system includes a production
21 file, and read-only snapshot copies of the production file, wherein the production file and
22 the read-only snapshot copies of the production file are organized as a version set
23 including an inode for the production file, an inode for each read-only snapshot copy of

1 the production file, and a set of file blocks including data blocks and indirect blocks that
2 are shared among the production file and the read-only snapshot copies of the production
3 file,

4 wherein the file server is programmed to maintain for each block in each snapshot
5 copy of the production file an indication of whether or not said each snapshot copy of the
6 production file is an oldest snapshot copy of the production file including an identical
7 version of said each block, and

8 wherein the file server is programmed to delete a read-only snapshot copy of the
9 production file, and when deleting the read-only snapshot copy of the production file, to
10 keep each block for which the read-only snapshot copy is not indicated as being an oldest
11 snapshot copy of the production file including an identical version of said each block.

12

13 31. The file server as claimed in claim 30, wherein each inode of each read-only
14 snapshot copy of the production file is linked to a hierarchy of blocks included in said
15 each read-only snapshot copy of the production file, and wherein the file server is further
16 programmed, upon deleting the read-only snapshot copy of the production file, to keep all
17 descendants of said each block for which the read-only snapshot copy is not indicated as
18 being an oldest snapshot copy of the production file including an identical version of said
19 each block.

20

21 32. The file server as claimed in claim 30, which is further programmed, upon
22 deleting the read-only snapshot copy of the production file, to keep each block for which
23 the read-only snapshot is indicated as being an oldest snapshot copy of the production file

1 including an identical version of said each block and a next-most recent snapshot copy of
2 the production file is indicated as not being an oldest snapshot copy of the production file
3 including an identical version of said each block.
4

5 33. The file server as claimed in claim 32, which is further programmed, upon
6 deleting the read-only snapshot copy of the production file, to indicate that the next-most
7 recent snapshot copy of the production file has become an oldest snapshot copy of the
8 production file including an identical version of said each block for which the read-only
9 snapshot is indicated as being an oldest snapshot copy of the production file including an
10 identical version of said each block and a next-most recent snapshot copy of the
11 production file is indicated as not being an oldest snapshot copy of the production file
12 including an identical version of said each block.
13

14 34. The file server as claimed in claim 30, which is further programmed, upon
15 deleting the read-only snapshot copy of the production file, to deallocate each block for
16 which the read-only snapshot copy is indicated as an oldest snapshot copy of the
17 production file including an identical version of said each block and a next most recent
18 snapshot copy of the production file is also indicated as an oldest snapshot copy of the
19 production file including an identical version of a block corresponding to said each block,
20 wherein said each block and the block corresponding to said each block are mapped to
21 the same logical file addresses.
22

1 35. A file server comprising storage containing a file system, and a processor coupled
2 to the storage for accessing the file system, wherein the file system includes a production
3 file, and snapshot copies of the production file, wherein the production file and the
4 snapshot copies of the production file are organized as a version set including an inode
5 for the production file, an inode for each snapshot copy of the production file, and a set of
6 file blocks including data blocks and indirect blocks that are shared among the production
7 file and the snapshot copies of the production file,

8 wherein the file server is programmed for responding to a request to create a read-
9 only snapshot copy of the production file by reserving for the production file a number of
10 free file blocks of at least the number of blocks in the production file.

11
12 36. The file server as claimed in claim 35, which is programmed for responding to a
13 request to create a read-write snapshot copy of a specified file in the version set by
14 reserving for the read-write snapshot copy a number of free blocks of at least the number
15 of blocks in the specified file in the version set.

16
17 37. The file server as claimed in claim 35, which is programmed for responding to a
18 request to restore the production file with a specified snapshot copy of the production file
19 by reserving for the production file a number of free blocks of at least the difference
20 between the number of blocks in the specified snapshot copy of the production file and
21 the number of blocks in the production file.

22

1 38. A file server comprising storage containing a file system, and a processor coupled
2 to the storage for accessing the file system, wherein the file system includes a production
3 file, and snapshot copies of the production file, wherein the production file and the
4 snapshot copies of the production file are organized as a version set including an inode
5 for the production file, an inode for each snapshot copy of the production file, and a set of
6 file blocks including data blocks and indirect blocks that are shared among the production
7 file and the snapshot copies of the production file,

8 wherein the file server is programmed for restoring the production file with a
9 specified snapshot copy of the production file by responding to a request to prepare to
10 restore the production file by preparing to restore the production file and reporting
11 whether or not preparation is successful, and then responding a request to commit the
12 preparation by restoring the production file with the specified snapshot copy of the
13 production file.

14
15 39. The file server as claimed in claim 38, which is programmed to prepare to restore
16 the production file by creating a read-write snapshot copy of the specified snapshot copy
17 of the production file, and which is programmed to commit the preparation by replacing
18 the production file with the read-write snapshot copy of the specified read-only snapshot
19 copy of the production file and deleting the production file, so that the read-write
20 snapshot copy of the specified snapshot copy of the production file assumes the identity
21 of the production file.

22

1 40. A file server comprising storage containing a file system, and a processor coupled
2 to the storage for accessing the file system, wherein the file system includes a production
3 file, and snapshot copies of the production file, wherein the production file and the
4 snapshot copies of the production file are organized as a version set including an inode
5 for the production file, an inode for each snapshot copy of the production file, and a set of
6 file blocks including data blocks and indirect blocks that are shared among the production
7 file and the snapshot copies of the production file,

8 wherein the file server is programmed for refreshing a specified snapshot copy of
9 the production file by creating a new inode in the version set, copying contents of the
10 inode of the specified snapshot copy into the new inode so that the new inode references
11 blocks of the specified snapshot copy, using the inode of the specified snapshot copy to
12 create a new snapshot copy of the production file by copying contents of the inode of the
13 production file into the inode of the specified snapshot copy, and performing a file
14 deletion upon the new node.

15

16 41. The file server as claimed in claim 40, wherein the file deletion upon the new
17 inode is performed asynchronously after using the inode of the specified snapshot copy to
18 create a new snapshot copy of the production file by copying contents of the inode of the
19 production file into the inode of the specified snapshot copy.

20

21 42. A method of operating a file server, the file server including storage containing a
22 file system, and a processor coupled to the storage for accessing the file system, wherein
23 the file system includes a production file, and read-only snapshot copies of the production

1 file, wherein the production file and the read-only snapshot copies of the production file
2 are organized as a version set including an inode for the production file, an inode for each
3 read-only snapshot copy of the production file, and a set of file blocks including data
4 blocks and indirect blocks that are shared among the production file and the read-only
5 snapshot copies of the production file, wherein the method comprises:

6 maintaining for each block in each snapshot copy of the production file an
7 indication of whether or not said each snapshot copy of the production file is an oldest
8 snapshot copy of the production file including an identical version of said each block, and

9 deleting a read-only snapshot copy of the production file, wherein the deleting of
10 the read-only snapshot copy of the production file includes keeping each block for which
11 the read-only snapshot copy is not indicated as being an oldest snapshot copy of the
12 production file including an identical version of said each block.

13

14 43. The method as claimed in claim 42, wherein each inode of each read-only
15 snapshot copy of the production file is linked to a hierarchy of blocks included in said
16 each read-only snapshot copy of the production file, and wherein the deleting of the read-
17 only snapshot copy of the production file includes keeping all descendants of said each
18 block for which the read-only snapshot copy is not indicated as being an oldest snapshot
19 copy of the production file including an identical version of said each block.

20

21 44. The method as claimed in claim 42, wherein the deleting of the read-only
22 snapshot copy of the production file further includes keeping each block for which the
23 read-only snapshot is indicated as being an oldest snapshot copy of the production file

1 including an identical version of said each block and a next-most recent snapshot copy of
2 the production file is indicated as not being an oldest snapshot copy of the production file
3 including an identical version of said each block.
4

5 45. The method as claimed in claim 44, which further includes, upon deleting the
6 read-only snapshot copy of the production file, indicating that the next-most recent
7 snapshot copy of the production file has become an oldest snapshot copy of the
8 production file including an identical version of said each block for which the read-only
9 snapshot is indicated as being an oldest snapshot copy of the production file including an
10 identical version of said each block and a next-most recent snapshot copy of the
11 production file is indicated as not being an oldest snapshot copy of the production file
12 including an identical version of said each block.
13

14 46. The method as claimed in claim 42, wherein the deleting of the read-only
15 snapshot copy of the production file includes deallocating each block for which the read-
16 only snapshot copy is indicated as an oldest snapshot copy of the production file
17 including an identical version of said each block and a next most recent snapshot copy of
18 the production file is also indicated as an oldest snapshot copy of the production file
19 including an identical version of a block corresponding to said each block, wherein said
20 each block and the block corresponding to said each block are mapped to the same
21 logical file addresses.
22

1 47. A method of operating a file server, the file server including storage containing a
2 file system, and a processor coupled to the storage for accessing the file system, wherein
3 the file system includes a production file, and snapshot copies of the production file,
4 wherein the production file and the snapshot copies of the production file are organized
5 as a version set including an inode for the production file, an inode for each snapshot
6 copy of the production file, and a set of file blocks including data blocks and indirect
7 blocks that are shared among the production file and the snapshot copies of the
8 production file, wherein the method comprises:

9 the file server responding to a request to create a read-only snapshot copy of the
10 production file, and when responding to the request to create a read-only snapshot copy
11 of the production file, reserving for the production file a number of free file blocks of at
12 least the number of blocks in the production file.

13

14 48. The method as claimed in claim 47, which includes the file server responding to a
15 request to create a read-write snapshot copy of a specified file in the version set, and
16 when responding to the request to create the read-write snapshot copy of the specified file
17 in the version set, reserving for the read-write snapshot copy a number of free blocks of
18 at least the number of blocks in the specified file in the version set.

19

20 49. The method as claimed in claim 47, which includes the file server responding to a
21 request to restore the production file with a specified snapshot copy of the production
22 file, and when responding to the request to restore the production file with the specified
23 snapshot copy of the production file, by reserving for the production file a number of free

1 blocks of at least the difference between the number of blocks in the specified snapshot
2 copy of the production file and the number of blocks in the production file.

3

4 50. A method of operating a file server, the file server including storage containing a
5 file system, and a processor coupled to the storage for accessing the file system, wherein
6 the file system includes a production file, and snapshot copies of the production file,
7 wherein the production file and the snapshot copies of the production file are organized
8 as a version set including an inode for the production file, an inode for each snapshot
9 copy of the production file, and a set of file blocks including data blocks and indirect
10 blocks that are shared among the production file and the snapshot copies of the
11 production file, wherein the method includes:

12 the file server restoring the production file with a specified snapshot copy of the
13 production file by responding to a request to prepare to restore the production file by
14 preparing to restore the production file and reporting whether or not preparation is
15 successful, and then responding a request to commit the preparation by restoring the
16 production file with the specified snapshot copy of the production file.

17

18 51. The method as claimed in claim 50, which includes the file server preparing to
19 restore the production file by creating a read-write snapshot copy of the specified
20 snapshot copy of the production file, and which includes the file server committing the
21 preparation by replacing the production file with the read-write snapshot copy of the
22 specified snapshot copy of the production file and deleting the production file, so that the

1 read-write snapshot copy of the specified snapshot copy of the production file assumes
2 the identity of the production file.

3

4 52. A method of operating a file server, the file server including storage containing a
5 file system, and a processor coupled to the storage for accessing the file system, wherein
6 the file system includes a production file, and snapshot copies of the production file,
7 wherein the production file and the snapshot copies of the production file are organized
8 as a version set including an inode for the production file, an inode for each snapshot
9 copy of the production file, and a set of file blocks including data blocks and indirect
10 blocks that are shared among the production file and the snapshot copies of the
11 production file, wherein the method comprises:

12 the file server refreshing a specified snapshot copy of the production file by
13 creating a new inode in the version set, copying contents of the inode of the specified
14 snapshot copy into the new inode so that the new inode references blocks of the specified
15 snapshot copy, using the inode of the specified snapshot copy to create a new snapshot
16 copy of the production file by copying contents of the inode of the production file into the
17 inode of the specified snapshot copy, and performing a file deletion upon the new node.

18

19 53. The method as claimed in claim 52, which includes the file server performing the
20 file deletion upon the new inode asynchronously after using the inode of the specified
21 read-only snapshot copy to create the new snapshot copy of the production file by
22 copying contents of the inode of the production file into the inode of the specified
23 snapshot copy.